VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

REZERVAČNÝ A VÝDAJNÝ SYSTÉM PRÍPRAVKOV PRE KALIBRÁCIU MIKROSKOPOV

RESERVATION AND DISPATCHING SYSTEM FOR MICROSCOPE CALIBRATION EQUIPMENT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MATEJ SLAHUČKA**
AUTHOR

VEDOUCÍ PRÁCE — **RNDr. MAREK RYCHLÝ, Ph.D.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/20267/2016/xslahu01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Slahučka Matej**
Obor: Informační technologie
Téma: **Rezervační a výdejní systém přípravků pro kalibraci mikroskopů**
Reservation and Dispatching System for Microscope Calibration Equipment
Kategorie: Databáze

Pokyny:

1. Seznamte se s problematikou výdeje přípravků pro kalibraci el. mikroskopů ve společnosti FEI Czech Republic. Analyzujte nejen samotný výdej, ale i návaznosti na související procesy/workflow.
2. Navrhněte vlastní systém pro rezervaci, automatický výdej a sledování využití přípravků pro kalibraci mikroskopů. Systém bude při rezervaci a výdeji analyzovat dosavadní použití přípravků (např. upozorňovat na potřebné revize) a jejich plánované použití (např. upřednostňovat výdej pro prioritní zakázky). Při výdeji a navrácení bude systém ovládat zámky přihrádek výdejního zařízení.
3. Po konzultaci s vedoucím systém implementujte.
4. Výslednou implementaci důkladně otestujte a pokud možno nasaďte do zkušebního provozu u dané společnosti.
5. Proveďte zhodnocení dosažených výsledků a diskutujte další možný vývoj projektu.

Literatura:

- Interní dokumentace společnosti FEI Czech Republic.
- Gamma, E.: *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1997, ISBN 0-201-63361-2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Rychlý Marek, RNDr., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Táto bakalárska práca sa zaoberá návrhom, implementáciou a testovaním informačného systému pre výdaj a rezerváciu prípravkov používaných na kalibráciu a nastavenie elektrónových mikroskopov a taktiež nadväznosťou na ďalšie procesy s tým súvisiace. Procesy sú popísane v úvodnej kapitole a podrobnejšie v časti venujúcej sa návrhu, sú tiež zahrnuté do finálnej implementácie systému. Systém bol vyvinutý na platforme .NET a pozostáva z troch individuálnych aplikácií.

Abstract

This bachelor thesis deals with an analysis, an implementation and a testing of a reservation and dispatching system for electron microscope calibration equipment. It explains the problem of reservations and also of dispatching with focus on related processes, which are needed for useful solution. These processes are discribed in the introduction and analysis chapter and included in the implementation of a system. The system is built with a help of .NET platform and consists of three individual applications.

Kľúčové slová

informačný systém, prípravok, výdaj, proces, rezervácia, .NET rámec, SQL

Keywords

information system, tool, dispatching, process, reservations .NET framework, SQL

Citácia

SLAHUČKA, Matej. *Rezervačný a výdajný systém prípravkov pre kalibráciu mikroskopov.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Rychlý Marek.

Rezervačný a výdajný systém prípravkov pre kalibráciu mikroskopov

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Mareka Rychlého. Ďalšie informácie mi poskytla spoločnosť FEI Czech Republic s.r.o. kde som pracoval. Uviedol som všetky literárne pramene a publikácie, z ktorý som čerpal.

.....
Matej Slahučka
15. mája 2017

Poďakovanie

Na tomto mieste by som rád poďakoval vedúcemu mojej práce, pánovi Markovi Rychlému za odborné rady, pomoc pri riešení problémov spojených s vypracovaním tejto práce a hlavne za čas, ktorý mi venoval.

Obsah

Kapitola 1

Úvod

Získavanie, ukladanie a následné analyzovanie informácií nadobúda v dnešnej dobe čoraz väčší význam. Informácie sa v súčasnosti stávajú strategickou veličinou, o ktoré sa je potrebné nielen starať z hľadiska nadobudnutia, ale aj z hľadiska ich spracovania. Veľmi často sa diskutuje o zavádzaní informačných systémov, výklad však nie je vždy jednoznačný. Úlohou prevádzky informačného systému je za pomoci moderných technológií zvýšiť efektivitu práce alebo ju uľahčiť a tým zlepšiť celkové výsledky firmy, či už priamo alebo nepriamo.

Cieľom tejto bakalárskej práce je navrhnúť, implementovať a otestovať rezervačný a výdajný systém prípravkov pre kalibráciu a nastavenie elektrónových mikroskopov.

Práca je rozčlenená do niekoľkých častí, ktoré predstavujú jednotlivé etapy pri vývoji informačných systémov. V úvodnej časti sú zadefinované základné pojmy. Pomocou procesov je vysvetlená problematika rezervácií a výdaja prípravkov vzhľadom k potrebám spoločnosti a vo všeobecnosti sú v nej zadefinované požiadavky, ktoré by mal nový systém spĺňať. V ďalšej časti je opísaný aktuálne používaný systém a sú tu tiež identifikované jeho hlavné obmedzenia. Nasledujúce kapitoly sa venujú návrhu nového informačného systému a jeho rozčleneniu, popisu implementácie hlavných funkcií a testovaniu. V záverečnej kapitole sú zhodnotené dosiahnuté výsledky, nasadenie informačného systému a tiež sú v nej diskutované ďalšie možnosti vývoja.

Kapitola 2

Úvod do problematiky

Cieľom tejto práce je navrhnúť, implementovať a otestovať rezervačný a výdajný systém prípravkov pre kalibráciu a nastavenie elektrónových mikroskopov. Systém má slúžiť, ako nástroj pre uľahčenie procesu výdaja prípravkov a vytvárania rezervácií. Musí tiež umožňovať analýzu využitia prípravkov a zohľadniť nadväznosť na súvisiace procesy.

Pod slovom prípravok budeme rozumieť prostriedok slúžiaci k splneniu určitej úlohy. Môže sa jednať o nástroj, prístroj alebo zariadenie.

Pre správne riešenie problému je nevyhnutné pochopenie používaných pojmov. Nasleduje stručné vysvetlenie.

2.1 Pojem informačný systém

Podľa [15] presná definícia pojmu *informačný systém* neexistuje a ani ju nie je možné jednoducho vytvoriť, pretože každý užívateľ alebo vývojár používa rôznu terminológiu a zdôrazňuje iné aspekty. Môžeme však povedať, že informačný systém je možné chápať, ako systém vzájomne prepojených informácií a procesov, ktoré s týmito informáciami pracujú. Pričom pod pojmom procesy rozumieme funkcie, ktoré spracovávajú informácie do systému vstupujúce. Zjednodušene môže tvrdiť, že procesy sú funkcie zabezpečujúce zber, prenos, uloženie, spracovanie a distribúciu informácií. Pod pojmom informácie potom rozumieme dáta, ktoré slúžia hlavne pre rozhodovanie a riadenie v rozsiahlejšom systéme.

Do celkovej funkcie informačného systému sa tiež premieta nezanedbateľná položka okolia. Okolie IS tvoria všetky objekty, ktoré zmenou svojich vlastností ovplyvňujú samotný systém, vrátane objektov, ktoré menia svoje vlastnosti v závislosti na systéme.

Celkovo teda môžeme povedať, že informačný systém je softvérové vybavenie firmy, ktoré je schopné na základe spracovávaných informácií riadiť procesy podniku, alebo poskytovať tieto informácie riadiacim pracovníkom tak, aby boli schopní vykonávať riadiace funkcie, medzi ktoré patrí plánovanie, koordinácia a kontrola všetkých procesov firmy.

2.2 Pojem proces

Proces môžeme definovať, ako štruktúrovanú a merateľnú sadu aktivít navrhnutých k vytváraniu konkrétneho výstupu pre určitého zákazníka na trhu. To zahrňuje silný dôraz na spôsob výkonu práce v danej firme, v protiklade k produktovému zameraniu, sústredenému na to, čo sa vykonáva. Proces je teda konkrétne usporiadanie aktivít v čase a priestore, s jasne definovaným začiatkom a koncom a jednoznačne určenými vstupmi a výstupmi: je

to štruktúra činnosti. Procesy sú teda štruktúry, pomocou ktorých firma robí to, čo je nutné na vytvorenie požadovaných hodnôt pre zákazníka [4].

2.3 Procesy spojené s využívaním prípravkov

Využívanie prípravkov definujú nasledujúce procesy. Ich identifikácia je nevyhnutná pre správny návrh a implementáciu nového systému.

2.3.1 Požičanie prípravku

Predstavuje hlavný proces a tvorí základ systému. Zamestnanec si vyberie prípravok, ktorý si chce požičať na základe potrieb konkrétnej zákazky na ktorej pracuje. Do jeho rozhodovania nevstupuje žiadna ďalšia osoba – pracuje úplne samostatne. Nevyhnutným predpokladom procesu je dostupnosť požadovaného prípravku a platnosť jeho revízie.

Výpožička musí obsahovať meno zamestnanca, ktorý ju vykonal, prípravok na ktorý sa vzťahuje, dátum a čas jej vzniku a pracovnú pozíciu zákazky.

Začiatok procesu teda predstavuje rozhodnutie zamestnanca požičať si prípravok. *Koniec procesu* je, že zamestnanec má prípravok požičaný a môže ho používať. Prípravok sa tým stáva nedostupný pre ostatných zamestnancov.

2.3.2 Vrátenie prípravku

Tento proces nadväzuje na proces požičania prípravku a uzatvára výpožičku, zmenou jej stavu na ukončený a doplnením dátumu a času vrátenia. Ak sa prípravok počas používania pokazil alebo bol poškodený má zamestnanec povinnosť to nahlásiť zodpovednej osobe, v takom prípade vznikne nová výpožička s atribútom porucha, pôvodná je uzavretá.

Začiatok procesu predstavuje rozhodnutie zamestnanca požičaný prípravok vrátiť. *Koniec procesu* je, že prípravok je vrátený a tým sa stáva opäť dostupný. Proces môže vykonať ľubovolný zamestnanec. Zodpovednosť za prípravok však nesie zamestnanec, ktorý proces požičania zahájil.

2.3.3 Vytvorenie rezervácie

Zamestnanec si vyberie prípravok, ktorý si chce zarezervovať. Zadá časový interval rezervácie, pracovnú pozíciu a prioritu zákazky na ktorej pracuje. Nakoniec si prípravok rezervuje.

Začiatok procesu predstavuje rozhodnutie zamestnanca vytvoriť rezerváciou. *Koniec procesu* je, že rezervácia bola vytvorená.

2.3.4 Výpožička z rezervácie

Zamestnanec si vyzdvihne prípravok na základe rezervácie. Rezervácia je tým ukončená a vzniká výpožička. V prípade, že zamestnanec prípravok nevráti v čase uvedenom v rezervácii je na to upozornený formou E-mailu.

2.3.5 Ukončenie rezervácie

Rezervácia je ukončená v momente keď je z nej vytvorená výpožička.

2.3.6 Expirácia rezervácie

Expiráciu rezervácie zabezpečuje systém na základe nastavených pravidiel, podľa nich je možné vyzdvihnúť prípravok maximálne s 30 minútovým meškaním. Na zrušenie je zamestnanec upozornený formou E-mailu. Expirovaná rezervácia nemôže byť obnovená.

2.3.7 Zrušenie rezervácie

Zamestnanec si zobrazí svoje rezervácie, ktoré sú usporiadané v prehľadnej tabuľke. Tá obsahuje základné informácie o rezervácii a tlačítka detail a zrušiť. Zrušená rezervácia nemôže byť obnovená.

2.3.8 Procesy správy systému

Procesy nevyhnutné pre správu systému zabezpečuje administrátor (správca systému). Medzi ne patrí pridávanie a odoberanie prípravkov, registrácia nových užívateľov, sledovanie platnosti revízií a prípadne riešenie porúch prípravkov. Správca nezabezpečuje ani nekontroluje výdaj prípravkov alebo vytvorené rezervácie. Má však možnosť ich editovať na požiadanie zamestnanca a tak opraviť prípadné chyby alebo záznamy zmazať.

2.3.9 Nadväzujúce procesy

Procesy riešenia porúch a zabezpečovania platnosti revízií nie sú ďalej podrobne rozpracované z dôvodu, že do výdaja prípravkov vstupujú iba okrajovo. Jedná sa o procesy, ktoré naväzujú na procesy výdaja. Ich podpora v systéme je zabezpečená možnosťou špecifikovať typ výpožičky a pomocou automatického upozorňovania správcu na blížiace sa revízie. Sýstém tým poskytuje nástroje pre uľahčenie týchto procesov a ich spustenie v správnom čase.

2.4 Zhodnotenie procesov

Opísané procesy jasne ukazujú, že systém je založený na samostatnosti a zodpovednosti každého zamestnanca a poskytuje mu veľký priestor pre vlastné rozhodnutia s minimálnymi obmedzeniami. Pre jeho správne fungovanie je však nevyhnutná vyššia miera firemnej kultúry.

Predstavuje tak modernú alternatívu k systémom s centrálnou autoritou napr. v podobe ďalšieho zamestnanca, ktorý by používanie prípravkov nejakým spôsobom plošne zastrešoval a informačný systém by využíval iba ako ďalší nástroj pri svojej práci. Návrh takéhoto systému nie je predmetom tejto práce, ani z jeho predpokladov nevychádza.

2.5 Stanovenie kľúčových požiadaviek

Pri vývoji softvéru vzniká často tendencia implementovať funkcionalitu, ktorá nie je potrebná alebo o ňu nie je záujem. Identifikáciu procesov spojených s riešením problémom je tomu možné predísť. Analýzou procesov 2.3 a zistených nedostatkov existujúceho systému 3.2 je možné stanoviť kľúčové požiadavky na nový systém, ktoré musí do úvahy vziať už samotný návrh. Medzi ne patrí:

- zabezpečenie jednoduchého a efektívneho užívateľského rozhrania,

- poskytnutie nástroja pre analýzu využívania prípravkov,
- zvýšenie komfortu pre zamestnancov,
- periodický audit revízií prípravkov a stavu rezervácií,
- jednotný formát záznamov o výpožičkách a rezerváciach.

Kapitola 3

Existujúce riešenie

V tejto kapitole je popísaný počiatočný systém výdaja a rezervácie prípravkov.

3.1 Popis

Bol používaný systém, ktorý sa skladal zo záznamových listov uložených na výdajnom mieste. Tie obsahovali tabuľky s prípravkami. Požičať si prípravok znamenalo vytvoriť záznam na patričnom riadku tabuľky, vrátiť prípravok naopak znamenalo daný záznam zmazať. Zapisovalo sa meno zamestnanca, dátum a čas požičania a pracovná pozícia.

3.2 Obmedzenia a nevýhody

Tento systém má množstvo nevýhod, ktoré značne obmedzujú jeho využiteľnosť a prínos v praxi. Ich identifikovanie je kľúčové pre návrh nového systému. Medzi hlavné obmedzenia patrí:

- Nemožnosť vzdialene zistiť stav jednotlivých prípravkov. Zamestnanec musí vždy opustiť svoje pracovisko, ísť na výdajné miesto, kde môže následne zistiť, že prípravok, ktorý potrebuje je momentálne nedostupný. Jedná sa o značnú neefektívnosť a plytvanie časom.
- Nie je možné sledovať históriu výpožičiek alebo rezervácií a tým pádom ani analyzovať ich používanie a reagovať na prípadne potreby.
- Systém nemá žiadny mechanizmus pre zabránenie odnesenia prípravku bez riadneho vyplnenia záznamového listu, čo môže spôsobovať zložité spätné dohľadávanie v prípade, že prípravok nie je na svojom mieste, záznam neexistuje a daný prípravok potrebuje niekto iný.
- Nepraktické mazanie záznamu pri vracaní prípravku. V prípade, že je záznamový list plný hrozí zmazanie aj okolitých záznamov.
- Veľmi obmedzená možnosť vytvárania rezervácií. Záznamy sa stávajú rýchlo neprehľadné.
- Náročná sumarizácia stavu prípravkov. Nutnosť prechádzať všetky záznamové listy alebo postupne jednotlivé uloženia.

- Po pridaní alebo odobraní prípravkov je nutné vždy vytvoriť a vytlačiť nový záznamový list.
- V prípade, že zamestnanec zabudne po vrátení prípravku zmazať záznam o výpožičke hrozí, že pri ďalšom požičiavaní sa bude prípravok podľa záznamového listu javiť ako nedostupný. Môže tak vzniknúť situácia, že sa čaká na prípravok, ktorý je v skutočnosti dostupný.
- Pre overenie platnosti revízie je nutné preskúmať revízny štítok, ktorý musí byť umiestnený na prípravku. Pre tento účel je nutné viesť ďalšie záznamy pracovníkom zodpovedným za revízie alebo kontrolovať prípravky každý deň.

Je očividné, že používaný systém je zastaralý a neefektívny a je potrebné vytvoriť nový systém. Nový systém musí eliminovať tieto obmedzenia alebo ich aspoň výrazne minimalizovať. Tiež si je možné všimnúť, že používaný systém nedodržiaval respektíve neposkytoval prostriedky pre procesy, ktoré systém výdaja definujú 2.3.

Kapitola 4

Analýza informačného systému

Vo fáze analýzy informačného systému je hlavným cieľom vytvoriť model, ktorý zachytáva hlavné požiadavky a charakteristiky budúceho informačného systému. Model špecifikuje akú funkcionalitu má systém pre užívateľov poskytovať, ale už nepopisuje akou technológiou potrebnú funkcionalitu zabezpečiť. Ústredným krokom analýzy je správne rozdelenie požiadaviek medzi jednotlivé podsystémy a zadefinovanie súvislostí medzi nimi. Snahou vždy je aby bol model čo najjednoduchší.

4.1 Modely životného cyklu softvéru

Životný cyklus systému sa skladá z jednotlivých etáp, ktoré určujú postupnosť krokov vývoja. Ďalej popisuje, akým spôsobom na seba etapy nadväzujú. Niektoré etapy sa môžu opakovať, prípadne aj čiastočne prekrývať. Cieľom je zabezpečiť aby výsledný systém spĺňoval zadané požiadavky.

Existuje veľké množstvo modelov, ktoré je možne rozdeliť do nasledujúcich kategórií. V praxi sa často využíva ich kombinácia [7].

Vodopádový model
: Vznikol v 70. rokoch minulého storočia a umožnil systematický vývoj softvéru. Jedná sa o sekvenčný model, ktorý rozdeľuje vývoj do samostatných, za sebou nasledujúcich etáp (Požiadavky, Návrh, Implementácia, Testovanie, Nasadenie a Údržba). Vyžaduje pevne stanovené požiadavky hneď na začiatku projektu, pretože ich neskoršia zmena má za následok návrat k predchádzajúcej etape. Z toho dôvodu je jeho využitie v praxi obmedzené. Pre vývoj informačných systémov kde je dodatočná zmena požiadaviek častá je úplne nevhodný.

Inkrementálny model
: Často označovaný aj ako prírastkový. Na začiatku sú zadané hrubé požiadavky. Systém je rozdelený na prírastky, ktoré sa vyvíjajú samostatne, postupne prípadne aj paralelne. Výhodou je, že po každom prírastku nasleduje overenie zo strany užívateľov. Nevýhodou je, že v prípade väčších projektov a častých zmien systém postupne degraduje.
Pre tento projekt bol zvolený práve inkrementálny model vývoja.

Evolučný model
: Je podobný inkrementálnemu modelu. Rozdiel je v tom, že definuje požiadavky na

začiatku každej iterácie. Predpokladom je, že systém sa neustále vyvíja na základe zmien požiadaviek zákazníka.

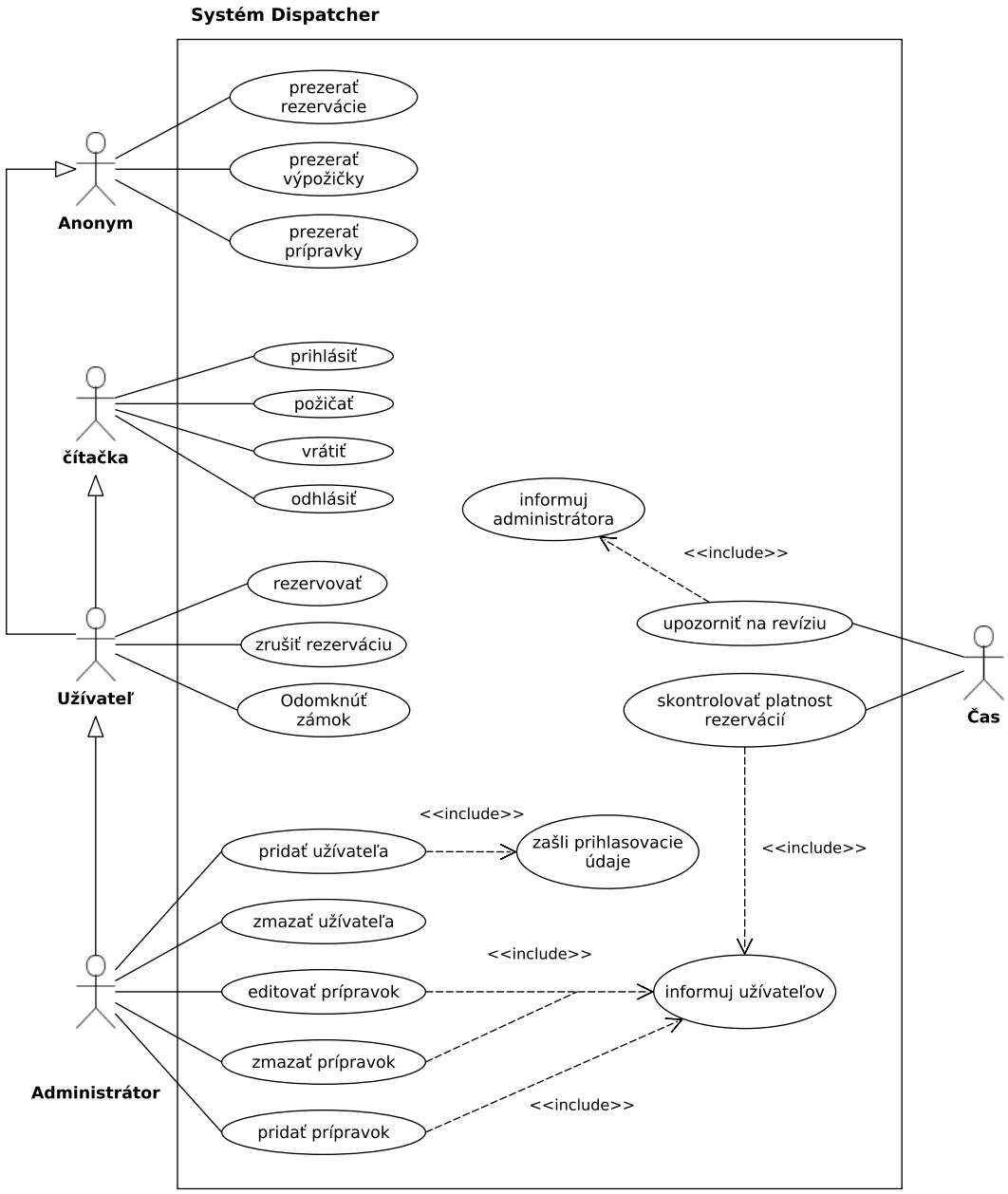
Špirálový model
Kombinuje prvky ostatných modelov. Hlavným princípom je neustále opakovanie niekoľkých etáp, až pokiaľ nie je systém hotový, pričom nové časti nadväzujú na predchádzajúci základ. Cyklus špirálového modelu pozostáva z nasledujúcich častí:

- určenie cieľov, alternatív a ohraničení;
- vyhodnotenie alternatív, identifikácia a riešenie rizík;
- vyvinutie, overenie produktu na ďalšej úrovni;
- plánovanie ďalších etáp.

4.2 Modelovanie prípadov použitia

Use Case Diagram – diagram prípadov použitia patrí do skupiny diagramov chovania jazyka UML. Používa sa k popisu systému z hľadiska užívateľa a zachytáva, ktoré typy užívateľov so systémom pracujú a aké činnosti v rámci systému vykonávajú respektíve môžu vykonávať [6]. Umožňuje znázorniť funkčné požiadavky na systém tým, že popisuje interakcie medzi ním a užívateľmi [2].

Vytvárajú ho prípady použitia, aktéri, vzťahy a hranica systému. Aktér popisuje užívateľa mimo systému, ktorý je s ním v interakcii. Týmto užívateľom nemusí byť iba fyzická osoba. Jeho názov vyjadruje jeho rolu v systéme. Vzťahy medzi aktérmi a prípadmi použitia znázorňujú medzi nimi plynúci tok informácií. Prípad použitia „špecifikuje časť funkcionality systému, ktorú využíva aktér a ktorá plní určitý cieľ“. Hranica systému je v modely znázornená rámčekom okolo prípadov použitia a názvom modelovaného systému [3].



Obr. 4.1: Diagram prípadov použitia

V nasledujúcej časti sú popísané kľúčové prípady použitia a ich význam v informačnom systéme.

Požičať

Proces požičiavania prípravkov rozoznáva dva spôsoby. *Fyzickú výpožičku* – to je taká, keď je prípravok uložený na výdajnom mieste a a takzvanú „*virtuálnu výpožičku*“ (vzdialenú), keď je prípravok už niekým požičaný t. j. nenachádza sa na výdajnom mieste. Dôvod pre zavedenie „virtuálneho“ požičiavania bol šetrenie času zamestnancov, zvýšenie využiteľnosti prípravkov. Hlavná motivácia spočívala v tom, že tým odpadá nutnosť neustále vracať prípravky na výdajné miesto aby vzápätí mohli byť

opäť vydané na pracovisko, ktoré je napr. na vedľajšej pozícii, ako pôvodné, z ktorého práve boli vrátené. Je však nevyhnutné aby tieto zmeny a presuny boli zaznamenané a bolo možné zistiť kde sa daný prípravok práve nachádza a kto ho má na starosti. Systém Dispatcher musí podporovať oba spôsoby výdaja a umožniť zamestnancom pohodlné a rýchle zmeny stavu prípravku.

Fyzickú výpožičku zabezpečuje desktopová aplikácia na výdajnom mieste, vzdialenú webová aplikácia.

Výpožička je definovaná svojím stavom a atribútmi. Stav môže byť aktívny (*Open*) alebo ukončený (*Closed*). Atribúty je možné rozdeliť na povinné a voliteľné. K povinným patrí: id zamestnanca, ktorý ju vykonal, id prípravku na ktorý sa vzťahuje, dátum a čas jej vzniku, pracovná pozícia zákazky a typ výpožičky (bežné použitie, porucha alebo revízia). K voliteľným atribútom patrí id rezervácie z ktorej prípadne výpožička vznikla a poznámka.

Vrátiť

Fyzicky vrátiť prípravok na výdajné miesto môže ktorýkoľvek zamestnanec. Stačí zosnímať čiarový kód a uloženie sa automaticky odomkne. Pri vzdialenom vracaní prípravku, musí zamestnanec využiť webovú aplikáciu. Výpožička sa ukončí a prípravok sa stáva opäť dostupný.

Rezervovať

Rezervácie vytvára zamestnanec. Rezervovať prípravky je možné od nasledujúceho dňa do dvoch týždňov. Maximálna dĺžka rezervácie sú tri dni. Zamestnanec si môže vyzdvihnúť zarezervovaný prípravok s maximálnym omeškaním 30 minút, inak jeho rezervácia automaticky vyprší a zamestnanec je na to upozornený formou E-mailu. V takom prípade si môže vytvoriť novú rezerváciu na najbližší možný termín alebo si prípravok požičať, ak bezprostredne nenasleduje ďalšia rezervácia. Ak zamestnanec zistí, že prípravok nie je dostupný napriek tomu, že bol riadne zarezervovaný. Systém mu o ňom zobrazí všetky dostupné informácie – výpožičku ktorej je súčasťou, kontaktné informácie na zamestnanca atď. Zrušenú ani vypršanú rezerváciu nie je možné znovu aktivovať alebo presunúť na iný termín.

Rezervácia je definovaná svojím stavom a atribútmi. Jej stav môže byť:

- Inquiry – aktívna, vytvorená;
- Canceled – zrušená užívateľom;
- Expired – rezervácia vypršala z dôvodu nečinnosti užívateľa, prípravok nebol vypožičaný;
- Closed – ukončená, prípravok bol vyzdvihnutý a beží výpožička.

K atribútom patrí: id zamestnanca, ktorý ju vykonal, id prípravku na ktorý sa vzťahuje, dátum a čas jej vzniku, časový interval pre ktorý je prípravok rezervovaný, pracovná pozícia a priorita zákazky (nízka, štandardná, vysoká). Systém tiež eviduje čas zmeny stavu rezervácie.

4.3 Členenie systému

Pre pokrytie potrebnej funkcionality je nutné systém rozdeliť do viacerých častí 5.3. Vzdialený prístup do systému z pracoviska zamestnanca bude zabezpečovať webová aplikácia. Pre

lokálny prístup a ovládanie zámkov v mieste výdaja prípravkov je potrebné vytvoriť desktopovú aplikáciu. Funkcie pre sledovanie revízií prípravkov a kontrolovanie stavov rezervácií sú periodický opakované. Predstavujú teda úlohy v čase. Z toho dôvodu ich nie je možné integrovať do ostatných aplikácií. Primárnou úlohou webového serveru je spracovávanie požiadaviek prostredníctvom HTTP protokolu, z hľadiska čistoty návrhu nie je teda správne ho používať aj ako plánovač úloh. Funkcie však nie je možné integrovať ani do desktopovej aplikácie. Úlohy by boli vykonávane iba v prípade, že aplikácia je neustále spustená. To však nie je možné zaručiť. Jediným možným riešením je vytvoriť tretiu aplikáciu a využiť služby poskytované operačným systémom (plánovač úloh) pre jej opakované spúšťanie.

4.4 Role užívateľov v systéme

Systém *Dispatcher* rozoznáva nasledujúce role užívateľov:

Anonymný užívateľ
: Je zamestnanec firmy, ktorý nie je prihlásený do systému. Má možnosť prezerať si stav jednotlivých prípravkov, výpožičiek a rezervácií.

Prihlásený užívateľ
: Je zamestnanec firmy prihlásený do systému, ktorý pre svoju prácu potrebuje pristupovať k prípravkom, tie si požičiava a následne vracia, prípadne vytvára rezervácie. Má prístup ku všetkým akciám anonymného užívateľa a tiež si môže prezerať štatistiky, ktoré systém poskytuje.

Administrátor
: Správca systému. Je zamestnanec, ktorý má na starosti prípravky a ich uloženie. Medzi jeho hlavné úlohy patrí pridávanie a odoberanie prípravkov, sledovanie ich stavu (poruchy, revízie) a využiteľnosti. V systéme tiež spravuje užívateľské účty. Zároveň má prístup ku všetkým akciám anonymného aj prihláseného užívateľa.

Čítačka
: Zariadenie na skenovanie čiarových kódov tzv. EAN kódy. Umožňuje prihlasovanie užívateľov do systému a slúži pre pohodlnú identifikáciu prípravkov.

Aplikácia DispatcherLocker (ovládanie zámkov) nepodporuje administrátorskú rolu z dôvodu nepraktickosti. Dotykové ovládanie nie je vhodné na spravovanie systému a tiež by tým bol zbytočne blokovaný výdaj pre ostatných užívateľov.

4.5 Návrh databázy

Databáza je súbor informácií, ako sú znaky, čísla, diagramy, ktorých systematická štruktúra umožňuje, aby tieto informácie mohli byť vyhľadávané pomocou počítača. Modelovať túto štruktúru je možné na viacerých úrovniach. Schému relačnej databázy je možné získať vytvorením konceptuálneho modelu a jeho následnou transformáciou na tabuľky relačnej databázy.

4.5.1 Entitne-relačný model

Najznámejšou a najčastejšie používanou modelovaciou technikou konceptuálneho návrhu relačných databáz je entitne-relačné modelovanie, jeho výsledkom je E-R diagram (ERD z anglického Entity-Relationship Diagram).

E-R diagram slúži k modelovaniu dát aplikačnej domény a a ich statických vzťahov. Je to sieťový model popisujúci návrh uložených dát v systéme na vyššej úrovni abstrakcie, reprezentuje teda logickú štruktúru databázy [9]. V tomto projekte bol konceptuálny model vytvorený práve pomocou E-R diagramu. Ten pozostáva z nasledovných prvkov [9]:

Entita
: Entita je „vec“ reálneho sveta, často býva označovaná ako objekt. Každá takáto entita musí byť odlíšiteľná od iných entít. Je popísaná hodnotami svojich atribútov.

Entitná množina
: Entitná množina je množina entít rovnakého typu, ktoré zdieľajú tie isté vlastnosti (atribúty). V dátovom modelovaní sa entity rovnakého typu klasifikujú do entitnej množiny, ktorá je následne zachytená v ERD.

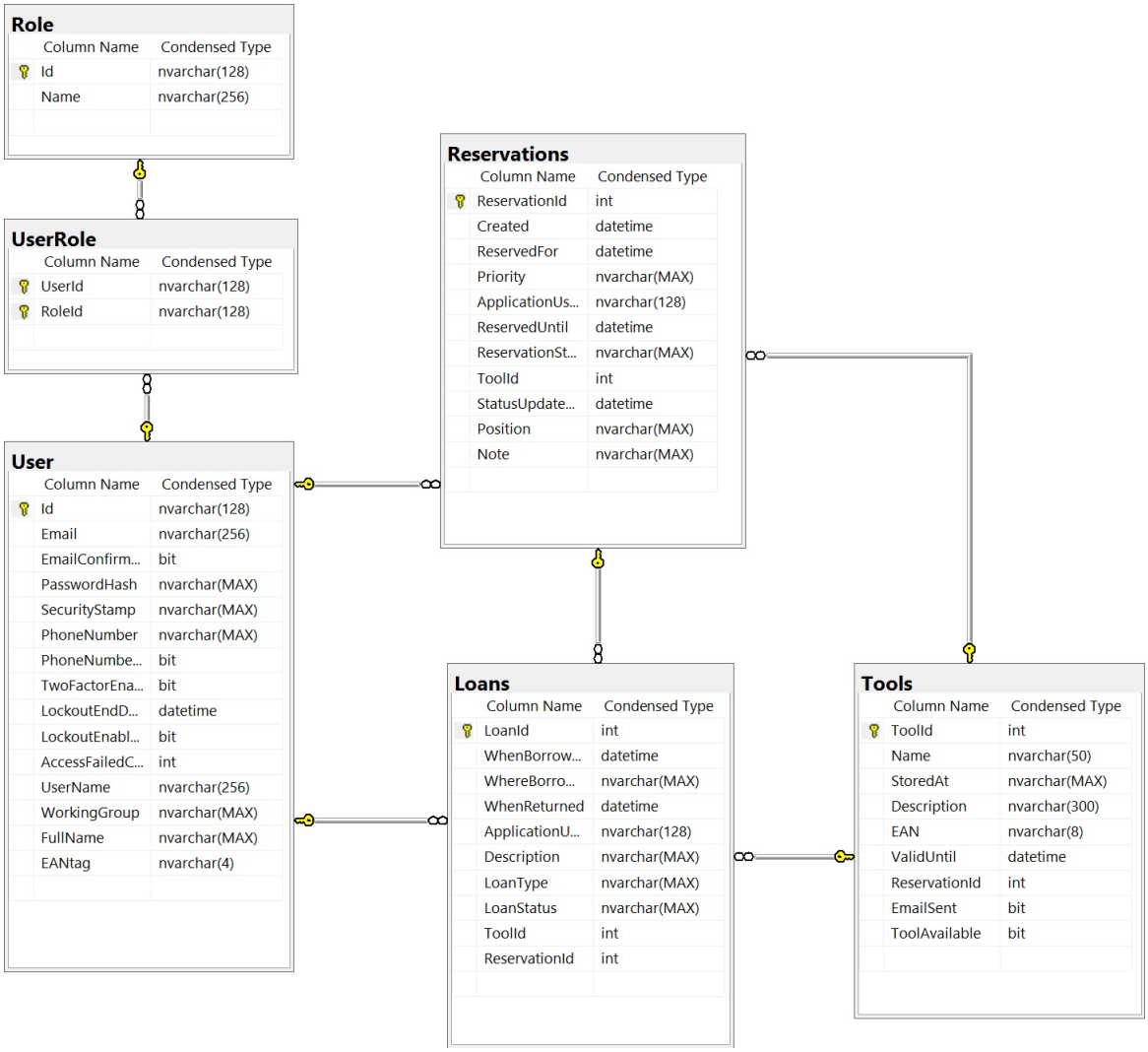
Vzťah
: Vzťah vyjadruje asociáciu medzi entitami, t. j. situáciu, kedy dve (alebo aj viacej) entity spolu logicky súvisia. Opisuje ho:

  - stupeň vzťahu – vyjadruje koľko entitných množín prislúcha do jednej vzťahovej množiny, najbežnejšie sa používajú vzťahy unárne (reflexívne) a binárne;
  - kardinalita – je maximálny počet vzťahov daného typu, v ktorých môže participovať jedna entita;
  - členstvo – vyjadruje nutnosť výskytu entity vo vzťahu t. j. minimálny počet vzťahov daného typu, v ktorých musí participovať jedna entita.

Vzťahová množina
: Vzťahová množina je množina vzťahov rovnakého typu alebo druhu.

Atribút
: Atribút je vlastnosť entity, ktorá nás v kontextu daného problému zaujíma. Atribút je definovaný ako prvok entitnej množiny.

Doména
: Doména atribútu predstavuje obor hodnôt, ktorý môže atribút nadobudnúť.



Obr. 4.2: Entity Relationship Diagram

Kapitola 5

Implementácia informačného systému

Implementácia je realizácia návrhu informačného systému a taktiež v sebe zahŕňa výber vhodných technológií.

5.1 Výber implementačných technológií

Pre implementáciu systému bolo zvolené prostredie *Microsoft Visual Studio 2015*[1] od spoločnosti *Microsoft*. Použité boli nasledujúce technológie:

1. Microsoft SQL Server 2016 – pre zabezpečenie dátovej vrstvy;
2. .NET 4 a jazyk C# 6 – pre aplikačnú a prezentačnú vrstvu.

5.2 Serverová časť

Serverovú časť tvorí databáza. Tá bola vytvorená pomocou nástroja *Entity Framework 6*[2] (ďalej len EF), týmto spôsobom boli zadefinované aj potrebné vzťahy. Medzi ne patrí určenie cudzích kľúčov, označenie primárnych kľúčov, vytvorenie indexov a ďalších nevyhnutných atribútov.

EF je ORM(z angl. Object-relational mapping) rámec pre ADO.NET. EF podporuje vývoj dátovo orientovaných aplikácií. Umožňuje vytvoriť tabuľky databázy a vzťahy medzi nimi z tried nadefinovaných v niektorom z jazykov platformy .NET. Tým dovoľuje vývojárom pracovať s dátami vo forme doménovo špecifických objektov, bez nutnosti znalosti konkrétneho rozloženia dát v tabuľkách a stĺpcoch ani presného spôsobu ich uloženia. Tento prístup poskytuje vyššiu mieru abstrakcie pri práci s dátami, čím umožňuje vytvárať a spravovať aplikácie kratším a prehľadnejším kódom. Keďže EF je súčasťou .NET, takto vytvorené aplikácie môžu byť používané na počítačoch s nainštalovaným .NET bez nutnosti inštalácie ďalšieho softvéru [10].

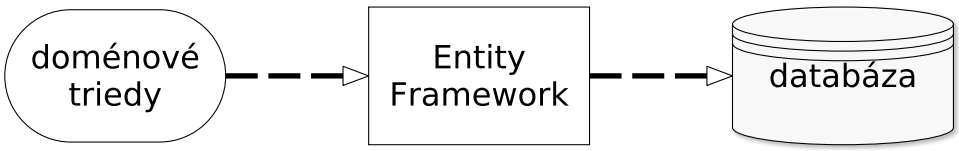
EF podporuje viacero prístupov pre vytvorenie databázy. V tomto projekte bol zvolený prístup *Code First*.

[1] viď: https://www.visualstudio.com/
[2] viď: https://github.com/aspnet/EntityFramework6

Model First
Tabuľky, vzťahy a hierarchia dedičnosti sa definujú pomocou grafického dizajnéra, vznikne tak model na základe ktorého je neskôr vytvorená databáza.

Database First
Z existujúcej databázy sú vygenerované príslušné triedy, s atribútami a hierarchiou dedičnosti.

Code First
V tomto prístupe sa vôbec nepracuje s grafickým dizajnérom. Vývojár najskôr vytvorí triedy, ktoré predstavujú tabuľky databázy. Tieto triedy nemôžu byť odvodené z iných špeciálnych podtried, ani nesmú obsahovať atribúty (agnlicky tzv. *properties*) vracajúce špeciálne typy. Následne pomocou dátových anotácií nastaví požadované vlastnosti atribútov, ktoré predstavujú stĺpce tabuľky. V rámci code first je silne využívaná paradigma konvencie pred konfiguráciou (anglicky *convention over configuration*)[3], čo zvyšuje čitateľnosť a flexibilitu zdrojového kódu. Vzťahy medzi tabuľkami sa vytvárajú pomocou kľúčového slova `virtual`.



Obr. 5.1: prístup Code First

5.3 Klientská časť

Klientskú časť tvoria tri aplikácie:

1. *DispatcherClient* – webová aplikácia, využívaná z pracoviska zamestnanca;
2. *DispatcherLocker* – desktopová aplikácia, spustená na počítači v mieste uloženia prípravkov;
3. *DispatcherChecker* – konzolová aplikácia, spustená na počítači v mieste uloženia prípravkov.

V ďalších častiach nasleduje popis jednotlivých aplikácií.

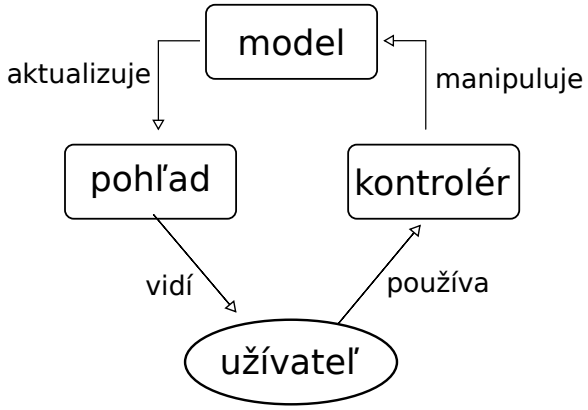
5.4 DispatcherClient

Jedná sa o webovú aplikáciu, ktorá je primárne využívaná zamestnancami a administrátorom systému. Táto aplikácia bola implementovaná pomocou aplikačného rámca *ASP.NET MVC*.

[3] viď: https://facilethings.com/blog/en/convention-over-configuration

5.4.1 Návrhový vzor

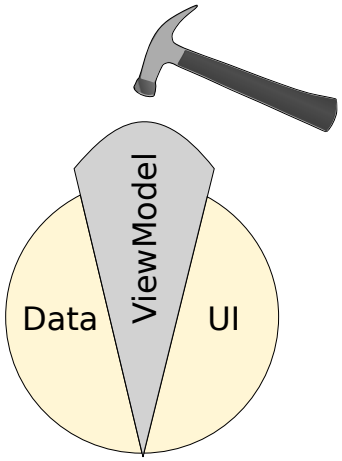
MVC (z angl. *Model-View-Controller*) označuje návrhový vzor, ktorý ASP.NET implementuje. Bol uvedený už v 70. rokoch 20. storočia. MVC rozdeľuje aplikáciu na tri logické vrstvy tak, aby bola možná ich prípadná úprava s minimálnym vplyvom na ostatné vrstvy. Tieto tri vrstvy sú Model (model), View (pohľad) a Cotroller (kontrolér). Model je zodpovedný za ukladanie a načítavanie dát, pohľad poskytuje grafické rozhranie pre interakciu s užívateľmi a kontrolér reaguje na udalosti (typicky vyvolané užívateľom) a zabezpečuje zmeny modelu alebo pohľadu [5].



Obr. 5.2: Diagram interakcií MVC

Podľa [10] hlavná nevýhoda architektúry MVC pri vývoji webových aplikácií je, že neposkytuje silne typovaný model pre pohľad.

Z toho dôvodu bol nakoniec, ako návrhový vzor zvolený modifikovaný MVC s pridaným *ViewModel* (model pre pohľad). Model pre pohľad poskytuje potrebné dáta pre pohľad. Je to ďalšia pridaná vrstva, ktorá sa nachádza sa medzi modelom a pohľadom. Kde zabezpečuje, že pohľad nemá prístup k celým doménovým entitám, ale iba k dátam v modely pre pohľad. Táto architektúra je označovaná, ako MVC VM alebo *MVC ViewModel* [10].



Obr. 5.3: Rola ViewModelu

Je potreba upozorniť, že význam a funkcia ViewModelu v architektúre MVC ViewModel nie je úplne totožná, ako v prezentačnom modely Model-View-ViewModel (MVVM), ktorý

je v dnešnej dobe používaný pri tvorbe desktopových aplikácií pomocou technológie WPF na systémoch Microsoft Windows.

5.4.2 Použité technológie

Pre vytváranie dynamických web stránok používa ASP.NET MVC značkovaciu syntax *Razor*, tá pozostáva z Razor značiek, C# kódu a HTML. Veľkou zvláštnosťou je možnosť používať jazyk C# na strane klienta [14].

Ďalej bol využitý *JavaScript* a formát *JSON*, knižnica *jQuery*[4]. Pre prácu k s databázou *LINQ to SQL* a pre obmedzenie častého znovu načítavania stránky bola použitá technológia *AJAX* (Asynchronous JavaScript and XML).

5.4.3 Smerovanie

ASP.NET umožňuje používať URL adresy, ktoré nemusia nevyhnutne mapovať špecifické súbory webovej stránky. Vďaka tomu je možné vytvárať URL adresy, ktoré obsahujú užitočné informácie a tým môžu užívateľom uľahčovať orientáciu. URL adresa ma typicky tvar `controller/action/id` [10].

5.4.4 Štruktúra aplikácie

Štruktúra aplikácie je do veľkej miery určená šablónou, ktorú generuje vývojové prostredie a je potrebné ju dodržiavať. Jedná sa o ďalší prejav prístupu konvencie pred konfiguráciou[3] rámca ASP.NET. K hlavným častiam patria adresáre: *Controllers* (obsahuje triedy implementujúce konroléry), *Models* (obsahuje triedy, ktoré reprezentujú doménové dáta celého systému), *ViewModels* (triedy, ktoré poskytujú potrebné dáta pre webové stránky, podmnožina doménových dát), *View* (webové stránky, rozhranie pre užívateľa).

K najdôležitejším konfiguračným súborom patria:

BundleConfig.cs
: Je súbor v adresári *App_Start* ktorý zabezpečuje minifikáciu *JavaScript* skriptov a zároveň umožňuje kombinovať súbory s obsahom *CSS* alebo JavaScript kódu na jednom mieste. Potrebnú funkcionalitu poskytuje trieda `BundleConfig` a jej metóda `RegisterBundles()`. V nej je možné vytvoriť akési skupiny podľa ich účelu a tie potom vložiť priamo do stránky, ako jeden celok pomocou metódy `Script.Render()`. Skripty sú automaticky minifikované, skombinované a načítané v jednom dotaze. Výsledkom je výrazné zlepšenie odozvy aplikácie na dotazy. [11]

  Určitou nevýhodou je, že skripty je potrebné vkladať v správnom poradí vzhľadom na ich vzájomné závislosti. To je však často náročné a prejavilo sa to aj v tomto projekte. Na túto nepríjemnosť je treba brať zreteľ aj v prípade rozširovania aplikácie o ďalšie funkcie.

Web.config
: Je hlavný konfiguračný súbor aplikácie, ktorý je uložený v koreňovom adresári projektu a musí byť validným XML dokumentom. V tomto súbore je možné zadefinovať viacero sekcií, týkajúcich sa rôznych oblastí napr. bezpečnosti, prekladu a ďalších. Jedna z najdôležitejších sekcií je nastavenie spojenia na databázový server (anglicky *connection string*), jej nastavenie je nevyhnutné pre správane fungovanie aplikácie.

[4] viď: https://jquery.com/

5.4.5 Implementácia

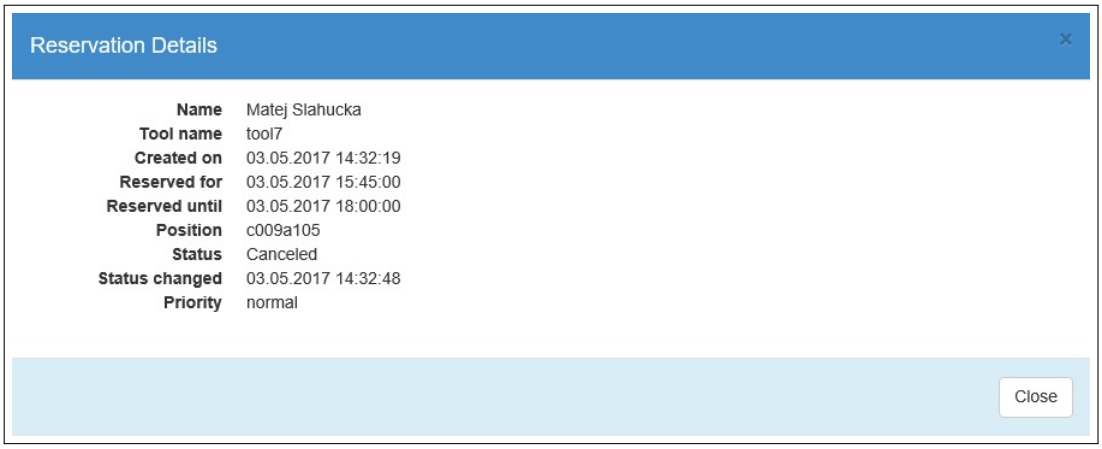
Aplikácia musí umožňovať užívateľovi navigáciu naprieč systémom. Jednotlivým akciám užívateľa zodpovedajú kontroléry, ktoré spravujú logiku aplikácie a stránky (vizualizácia dát) a spracovávajú požiadavky zadané užívateľom.

Ak užívateľ spustí akciu ku ktorej nemá autorizáciu, zo serveru príde zamietavá odpoveď (kód 401). To sa však pri bežnej práci v aplikácii nestane – aplikácia neposkytuje užívateľovi možnosť prístupu k ovládacím prvkom, ktoré sú nedostupné pre jeho rolu v systéme. Nie je však možné ošetriť manuálne zadanie URL cesty vedúcej na stav, ku ktorému užívateľ nemá prístup.

Pre zobrazenie detailov o výpožičkách a rezerváciach boli použité modálne okná rámca *Bootstrap* (príklad je možné vidieť na obrázku 5.4). Načítanie dát v tomto prípade prebieha asynchrónne pomocou `Ajax.ActionLink()`

Analýza využitia prípravkov a aktivity zamestnancov je dostupná vo forme histogramov. K sledovaným patrí: najviac požičiavaný a rezervovaný prípravok, priemerná doba výpožičky a aktivita jednotlivých zamestnancov. Pre implementáciu bola použitá knižnica *jqPlot* a *JavaScript*.

V dôsledku použitia rámca Bootstrap je aplikácia responzívna a je možné je používať na rôznych zariadeniach. To platí aj o stránke, ktorá zobrazuje histogramy. Tam však pre dosiahnutie tejto vlastnosti bolo nutné použiť knižnicu *jQuery*.



Obr. 5.4: Ukážka detailu rezervácie

5.5 DispatcherLocker

DispatcherLocker je desktopová aplikácia určená pre širokouhlý dotykový monitor. Je spustená na počítači v mieste uloženia prípravkov, kde poskytuje základnú funkcionalitu systému. Tam patrí:

- požičiavanie prípravkov;
- vracanie prípravkov;
- ovládanie zámkov;
- výdaj zarezervovaných prípravkov;

- poskytovanie základných informácií o dostupnosti prípravkov, výpožičkách a rezerváciach.

5.5.1 Použitá technológia

Z dôvodu, že sa jedná o aplikáciu určenú pre dotykový monitor, kde je požadovaná určitá strohosť v podobe malého počtu grafických prvkov, ich patričnej velikosti, umiestnenia atď. tak aby bolo ovládanie prstom pohodlné a presné. Bolo rozhodnuté, že aplikácia bude implementovaná s využitím grafickej knižnice *Windows Forms* (*WinForms*), ktorá je súčasťou platformy .NET. WinForms neposkytuje síce také možnosti konfigurovateľnosti rozhrania, ako modernejší grafický subsystém Windows Presentation Foundation (WPF), ale pre potreby tohoto projektu je úplne dostatočný.

Implementačný jazyk je opäť C#. Pre prácu s databázou bol použitý modul `System.Data.SqlClient`, ktorý poskytuje všetky potrebné funkcie, vrátane parametrizovateľných SQL dotazov.

5.5.2 Implementácia

Identifikácia užívateľov a prípravkov je zabezpečená pomocou EAN kódov (Code 128). Zadávanie je možné pomocou čítačky čiarových kódov alebo virtuálnej klávesnice. Dĺžka kódov je v oboch prípadoch pevne daná, takže spracovanie vstupu je z hľadiska programátora jednoduché, stačí zachytiť udalosť `TextChanged(object sender, EventArgs e)` a pri správnej dĺžke vstupu spustiť potrebnú obslužnú rutinu.

Po prihlásení vidí užívateľ všetky svoje aktívne výpožičky a rezervácie, ich zobrazenie a prácu s nimi zabezpečuje trieda `DataGridView`. Po kliknutí, udalosť `SelectionChanged( object sender, DataGridViewCellEventArgs e)`, na záznam sa mu automaticky odomkne uloženie, prípadne vytvorí výpožička z existujúcej rezervácie. Užívateľ má tiež možnosť vytvoriť úplne novú výpožičku.

Jeden z náročnejších problémov na implementovanie vychádzal z nutnosti zabezpečiť automatický prechod do predvoleného formuláru (loginForm.cs) v prípade, že užívateľ nedokončí všetky potrebné kroky práve vykonávanej akcie. Ako príklad je možné uviesť neaktivitu užívateľa po prihlásení. Riešenie spočívalo v zavedení časovača a v sledovaní signálov, ktoré zasiela operačný systém do aplikácie pri rôznych akciách užívateľa. V tomto prípade bolo potrebné sledovať signály `WM_KEYDOWN`, `WM_KEYUP`, `WM_SYSKEYDOWN` a `WM_SYSKEYUP`. Pri zmene stavu aplikácie z predvoleného formulára sa automaticky spustí na pozadí časovač, ak je zaslaný signál t. j. užívateľ je aktívny nastaví sa na pôvodnú hodnotu inak vyprší a užívateľ je odhlásený. Hodnota časovača je nastavená na dve minúty a bola odvodená z priemerného času, ktorý užívateľ strávi v aplikácii po prihlásení.

```
1.    timer._idleTimer = new Timer();
2.    timer._idleTimer.Interval = 1000 * 60 * 2;
3.    timer._idleTimer.Tick += (object s, EventArgs e) =>
        _idleTimer_Tick(s, e, relayControl);
4.    Application.AddMessageFilter(new KeyboardMessageFilter(UserIsActive));
```

Ukážka kódu 5.1: Nastavenie časovača a filtru

5.5.3 Štruktúra

Nasledujúce súbory implementujú hlavnú funkcionalitu aplikácie:

- *BoardConnectForm.cs*: Predstavuje vstupný bod aplikácie. Zabezpečuje detekciu USB relé karty[5].
- *LoginForm.cs*: Predvolený formulár aplikácie. Umožňuje prihlasovanie užívateľov a poskytuje základný prehľad o stave výpožičiek, rezervácií a prípravkoch pre anonymného užívateľa.
- *LockerControlForm.cs*: Zabezpečuje odomykanie uložení prípravkov a zobrazuje informácie pre prihláseného užívateľa, aktívne výpožičky a rezervácie, a umožňuje s nimi ďalej pracovať.
- *BorrowForm.cs*: Umožňuje požičiavanie prípravkov.
- *ReturnForm.cs*: Vracanie prípravkov.
- *ChoosePositionForm.cs*: Výber pracovnej pozície. Kam bude prípravok požičaný.
- *SerialInterface.cs*: Zabezpečuje vytvorenie sériového rozhrania a prácu s ním.
- *ModuleInterface.cs*: Poskytuje rozhranie pre ovládanie relé.

5.5.4 Systém ovládania zámkov

Aplikácia *DispatcherLocker* pri svojej činnosti pracuje s elektromagnetickými zámkami[6] (ďalej len zámok), ktoré ovláda pomocou relé. Použité zámky nie sú stavané na nepretržitú záťaž v podobe prechádzajúceho elektrického prúdu. Po určitom čase, približne po piatich minútach, by došlo ku skratu na cievke, v dôsledku vysokej teploty, a tiež k poškodeniu plastového obalu v ktorom je celý mechanizmus uložený. Zámok sa tým stáva nepoužiteľný a neopraviteľný, vzniká finančná škoda a výdajný systém musí až do výmeny poškodenej súčiastky fungovať v obmedzenom režime. Taktiež vzniká určité bezpečnostné riziko.

Preto je nevyhnutné zabezpečiť, aby k takýmto situáciám vôbec nedochádzalo, správnou prácou s USB Relé kartou[7]. Jediná možnosť riešenia je opäť použitie časovača a hneď po zatvorení relé (odomknutí zámku) zavolať funkciu pre otvorenie relé (zamknutie zámku). Vzniká tak dvojica funkcií, ktoré musia byť vždy použité spolu. Spoliehať sa, že užívateľ prácu s aplikáciou korektne ukončí by bola chyba, ktorá by mohla viesť k poškodeniu zámkov. Z toho dôvodu nie je možné pracovať s relé v rámci sekvenčného behu programu.

Komunikácia s kartou prebieha na sériovej zbernici USB (anglicky Universal Serial Bus) s využitím modulu `using System.IO.Ports` a funkcií `SerialCommPort.Send()` a `SerialCommPort.Read()`.

[5] Aplikáciu je možné používať aj bez pripojenej karty, stačí potvrdiť voľbou pri spustení. Samozrejme ovládanie zámkov nebude v takom prípade fungovať. Hlavná motivácia pre zavedenie tejto možnosti spočívala v snahe zabezpečiť vyššiu pružnosť systému v prípade porúch hardvéru.

[6] Boli použité zámky od firmy BERA, ktoré sa v Českej republike najčastejšie predávajú pod názvom BERA SZE 24 V AC/DC – Skříňový zámek elektrický STANDARD

[7] Použitý bol modul *EasyDaq USB4PRMxN*. Obsahuje štyri relé. Modul je dizajnovaný na zátaž do 240 V AC pri max. 10 A. Je napájaný priamo z USB. Čo je dostatočné iba na prepínanie relé bez akejkoľvek zátaže. Pre ovládanie ďalších komponentov je nutný prídavný napájací zdroj. Aplikácia bola vyvíjaná a testovaná iba s týmto modulom, jej funkčnosť s inými modelmi nebola vôbec overovaná. Pre ďalšie informácie viď: http://www.easydaq.biz/PagesUSB/USB4PRMxNFRAME.htm

5.6 DispatcherChecker

DispatcherChecker je konzolová aplikácia, ktorá zabezpečuje vykonávanie opakovaných úloh nad databázou. Patrí medzi ne kontrola platnosti revízií prípravkov a identifikácia vypršaných rezervácií.

5.6.1 Implementácia

Aplikácia podporuje zmenu nastavení bez nutnosti zásahu do zdrojového kódu. Konfigurácia pozostáva z nastavenia:

- spojenia s databázou (anglicky connection string),
- spojenia na SMTP (anglicky Simple Mail Transfer Protocol) server,
- tela E-mailu a príjemcu správ týkajúcich sa platnosti revízií,
- tela E-mailu pre správy ohľadom vypršaných rezervácií.

Nastavenia je možné meniť v XML súbore *DispatcherChecker.exe.config*. Zmeny sa automaticky prejavia pri nasledujúcom spustení programu. Pre implementáciu tohoto chovania bol použitý modul `System.Configuration` a trieda `ConfigurationManager`, ktorá obsahuje všetky potrebné metódy. Pre posielanie E-mailov modul `System.Net.Mail` a pre prácu s databázou modul `System.Data.SqlClient`.

Aplikácia nebola implementovaná ako služba do operačného systému z dôvodu ponechania väčšej miery konfigurovateľnosti pre správcu systému. Pomocou plánovača úloh (anglicky Windows Task Scheduler) je možné nastaviť časové intervaly bez nutnosti zmeny zdrojového kódu a následného prekladu. Toto riešenie nie je len flexibilnejšie, ale aj spoľahlivejšie ako použitie časovačov v `System.Timers`.

5.7 Poznámka k implementácii

Aplikácie sa vzájomne dopĺňajú a vytvárajú jednotný celok, ktorý zabezpečuje všetky funkcie kladené na systém. Takéto rozdelenie bolo nevyhnuté a určovali ho jednak požiadavky, povaha systému a tiež obmedzenie a určenie použitých technológií. Ako príklad je možné uviesť aplikáciu DispatcherChecker, ktorej funkcionalitu nebolo možné žiadnym spôsobom integrovať do ostatných aplikácií.

Pre prehľadnosť nasleduje krátke zhrnutie.

DispatcherLocker
: Desktopová aplikácia, určená pre širokouhlý dotykový monitor v mieste uloženia prípravkov, ktorá zabezpečuje základnú funkcionalitu systému. Primárne slúži na prihlasovanie užívateľov, výdaj a vracanie prípravkov pomocou čítačky čiarových kódov. Ďalej na ovládanie elektronických zámkov a taktiež poskytuje základný prehľad o dostupnosti prípravkov, stave výpožičiek a rezervácií.

DispatcherClient
: Webová aplikácia, ktorá poskytuje rozšírenú funkcionalitu systému. Jedná sa o napríklad funkcie pre administrátora, ako správa užívateľov a prípravkov, zadávanie porúch a revízií prípravkov, ale aj funkcionalitu pre bežných užívateľov, vytváranie rezervácií, podrobný prehľad výpožičiek a virtuálne požičiavanie a vracanie prípravkov.

DispatcherChecker
: Konzolová aplikácia, ktorá zabezpečuje upozorňovanie, formou E-mailu, správcu systému na blížiace sa technické revízie prípravkov a užívateľov, o zmene stavu rezervácie v prípade nevyzdvihnutia prípravku.

5.8 Poznámka k systému ovládania zámkov

Ani najlepší návrh a testovanie nedokáže zaručiť bezchybnosť a dostupnosť softvéru v každom okamihu. V prípade systémov ovládajúcich hardvér by vždy mala existovať možnosť zariadenie bezpečne vypnúť a pokiaľ je to možné vyradiť zo systému s minimálnym dopadom na funkcionalitu. Návrh a implementácia aplikácie DispatcherLocker tento všeobecný predpoklad spĺňuje a je možné ju používať aj bez USB relé karty.

Ďalší veľký problém môže nastať v prípade zlyhania systému ovládania zámkov. Všetky prípravky by sa stali pre zamestnancov okamžite nedostupné. Táto situácia môže vzniknúť z rôznych dôvodov. Napríklad porucha elektronickej súčiastky alebo ako dôsledok softvérovej chyby. Z toho dôvodu bol zavedený aj alternatívny, mechanický spôsob odomykania zámkov pomocou nástroja.

Podrobný popis, schéma systému a zoznam použitých elektronických súčiastok v tejto práci nie je uvedený z dôvodu, že táto práca je primárne zameraná na softvérové riešenie problému.

Kapitola 6

Testovanie informačného systému

Testovanie systému je neoddeliteľnou súčasťou vývoja a nasadenia systému do prevádzky. Zahŕňa v sebe odhaľovanie softvérových chýb, nedostatkov a nezrovnalostí v systéme. Pri zvolení správneho prístupu môže tieto problémy včas zachytiť a tým zamedziť neskorším komplikáciám, časovým a finančným stratám, ktoré by boli inak vynaložené na ich odstraňovanie.

V prípade zahrnutia užívateľov do procesu testovania, je možné tiež odhaliť nevhodný návrh užívateľského rozhrania alebo aplikačnej logiky systému. Pri testovaní tohoto systému sa kládol dôraz na užívateľov a ich požiadavky, ale aj na overenie individuálnych častí systému pomocou Unit testov.

6.1 Unit testy

Unit testovanie je metóda, ktorou sa overujú individuálne jednotky zdrojového kódu. To sú súbory jedného alebo viacerých modulov programu, spolu so súvisiacimi kontrolnými dátami, procedúrami použitia a operačnými procedúrami, tak aby sa zistilo, či sú vhodne zvolené [1]. Jedná sa o prvú fázu procesu testovania softvéru. V prípade objektovo orientovaných systémov tieto jednotky predstavujú triedy alebo metódy.

V ideálnom prípade, veškerá funkcionalita a všetky metódy by mali byť pokryté unit testami. To je v praxi nedosiahnuteľné, z dôvodu vysokej zložitosti systémov. Napriek tomu je stále možné dosiahnuť relatívne vysoké pokrytie kódu. Aby si vývojári mohli byť istý, že systém spĺňa požiadavky musia všetky jednotky testy splniť. Testovanie je automatické a vždy prebieha v súlade k tomu určenému rámcu. Pre tento projekt bol využitý testovací rámec *NUnit*[1]. Ten je určený pre programovacie jazyky patriace do platformy .NET.

Unit testy pre tento systém sú umiestnené v samostatnom projekte s názvom *DispatcherClient.Tests* a pokrývajú najpoužívanejšie a najkritickejšie funkcie v systéme, a to požičiavanie a vracanie prípravkov, vytváranie a rušenie rezervácií a ďalšie.

6.1.1 Obmedzenie Unit testov

Keďže unit testy vytvára programátor, ktorý je zároveň aj vývojárom celého projektu. Je možné namietnuť, že vo všeobecnosti nie je dobré, kontrolovať niečo sám po sebe. Z dôvodu, že keď niečo vytvoríte, tak iba zriedkakedy v tom aj odhalíte vlastné chyby.

[1] viď: https://www.nunit.org/

Na prekonanie tohoto obmedzenia a dôkladnejšie otestovanie systému bolo do testovania nutné zapojiť aj užívateľov.

6.2 Testovanie užívateľského rozhrania a aplikačnej logiky

Cieľom tohoto informačného systému je nahradiť aktuálne používaný systém a eliminovať všetky jeho nedostatky. Preto je nevyhnutné aby bolo užívateľské rozhranie intuitívne, jednoduché a poskytovalo prístup k potrebnej funkcionalite s minimálnym počtom kliknutí.

K hlavným používateľom systému patria pracovníci výroby. Vzorka z nich bola preto využitá na otestovanie užívateľského rozhrania a do určitej miery aj aplikačnej logiky. Testovanie prebiehalo s využitím prvého návrhu užívateľského rozhrania, ktoré im bolo poskytnuté spolu s jednoduchými úlohami. Pri ich plnení boli pozorovaní vývojárom systému.

Úlohy pre aplikáciu *DispatcherClient* boli nasledovné:

- požičanie a vrátenie prípravku,
- vytvorenie a zrušenie rezervácie,
- zistenie stavu zadaného prípravku,
- jednoduché vyhľadanie v histórií výpožičiek.

Úlohy pre aplikáciu *DispatcherLocker* boli nasledovné:

- požičanie a vrátenie prípravku,
- odomknutie uloženia,
- zistenie stavu prípravku.

Jednalo sa teda o hlavné funkcie tvoriace jadro systému, ktoré budú využívané najčastejšie. Užívatelia hneď pri prvom použití zvládli obsluhu a všetky úlohy dokončili. Mali však aj pripomienky a návrhy na zlepšenie.

6.2.1 Pripomienky k aplikácii DispatcherClient

Z pohľadu užívateľov bolo nevyhovujúce usporiadanie menu, ktoré v niektorých prípadoch obsahovalo zanorenie až do tretej úrovne. Bol zadaný návrh na jeho prepracovanie, ktorý mal jednu podmienku. Tá bola, že základná funkcionalita musí byť dostupná hneď po prihlásení bez potreby ďalšieho klikania a menej používaná maximálne v druhej úrovni.

Na základe tohoto návrhu bolo menu prepracované, základná funkcionalita bola presunutá na hlavnú stránku. Pričom boli využité dostatočne veľké tlačítka, logicky usporiadané do skupín, s glyf ikonami, tak aby bol hneď na prvý pohľad jasný ich účel.

6.2.2 Pripomienky k aplikácii DispatcherLocker

V prípade aplikácie *DispatcherLocker* sa užívateľom, ako nevyhovujúci javil postup pri vracaní prípravkov, teda problém bol skôr v aplikačnej logike, ako v užívateľskom rozhraní. Keďže systém je používaný v uzavretom prostredí, bolo navrhnuté aby, bolo možné vracať prípravky bez nutnosti prihlásenia, iba na základe zosnímania čiarového kódu.

Tento spôsob bol prijatý nakoľko rieši problém neprítomnosti zamestnanca, ktorý má niečo požičané, ale v daný deň nie je v práci, bez nutnosti zásahu administrátora – prípravok je tak k opäť dispozícii v kratšom čase. Takýto systém je bežne využívaný aj inde, napríklad v knižniciach. Je až zarážajúce, že rovnaký systém vracania prípravkov nebol použitý hneď v návrhu, čím sa potvrdil veľký význam zapojenia užívateľov do vývoja softvéru.

6.3 Testovanie systému ovládania zámkov

Popis implementácie systému ovládania zámkov, aj z pohľadu bezpečnosti, je uvedený v kapitole 5.5.4.

Z hľadiska času sa jednalo o najpracnejšiu časť testovania výdajného systému, keďže testy nebolo možné automatizovať. Pre overenie správnosti tohoto systému bolo nevyhnutné manuálne vykonať nasledujúce testy:

- v každom úseku kódu v ktorom sa relé zatvára, skontrolovať previazanosť s funkciou na asynchrónne otvorenie relé a tiež zmysluplnosť časovej hodnoty do nej predanej.
- overiť fungovanie časovača, ktorý zabezpečuje, že sa aplikácia vždy vráti do predvoleného formulára, `loginForm.cs`, bez ohľadu na správnosť akcií alebo nečinnosť užívateľa v niektorom kroku používania aplikácie.
- overiť, že v predvolenom formulári, `loginForm.cs`, sa relé vždy rozpoja po uplynutí určitého časového intervalu.
- skontrolovať reakcie na signály pre ukončenie aplikácie(zatvorenie okna, reštart počítača atď.) a ich príslušné spracovanie.

Je nutné upozorniť, že tieto kroky iba výrazne minimalizujú, že dôjde k poškodeniu zámku. Niektoré udalosti nie je možné softvérovo ošetriť. Napríklad násilné vypnutie počítača alebo zamrznutie operačného systému, prípadne aj neodhalené chyby v aplikácii *DispatcherLocker* by teoreticky mohli viesť k poškodeniu.

6.4 Testovanie podpory webových prehliadačov

Kedže *DispatcherClient* je webová aplikácia a pretože implementácia JavaScriptu je medzi jednotlivými webovými prehliadačmi často mierne odlišná bolo nutné overiť jej funkcionalitu v tých najbežnejších.

Internet Explorer 11
: Je doporučený prehliadač pre túto aplikáciu. Aplikácia bola primárne vyvíjaná pre tento webový prehliadač, pretože je výhradne používaný v spoločnosti FEI. Veškerá funkcionalita, z hľadiska prehliadača funguje správne. V prípade nejakých problémov sa s najväčšou pravdepodobnosťou jedná o chyby na strane aplikácie. Tie však môžu byť opravené.

Firefox 50
: Bola overená hlavná funkcionalita. Dynamičnosť aplikácie funguje správne. Neboli pozorované žiadne problémy.

Podpora v prehliadači *Google Chrome* nebola testovaná. Nie je však známy dôvod pre vznik inkompatibility alebo nefunkčnosti.

6.5 Zhodnotenie testovania

Testovaním nie je možné zaručiť bezchybnosť systému z dôvodu, že sa nedajú nasimulovať všetky vstupné hodnoty a následne skontrolovať správnosť výstupov. Testovanie je teda spôsob, ako poukázať na prítomnosť chýb, ale nie je možné takto dokazovať ich neprítomnosť [8]. Vždy sa budú objavovať chyby a problémy v produkčnom nasadení systému, ich zvládnutie a odstránenie však z hľadiska životného cyklu informačných systémov patrí už do fázy údržby. Tá môže byť výrazne uľahčená správnym návrhom a vypracovaním kvalitnej dokumentácie o všetkých častiach systému.

6.5.1 Poznámka k testovaniu užívateľského rozhrania

Testovanie užívateľského rozhrania priamo na užívateľoch má aj svoje nevýhody. Jednou z nich je možnosť ľahko skĺznuť do prostého zbierania názorov na použité grafické prvky a ich rozloženie, tvar, farbu atď. namiesto overenia použiteľnosti rozhrania a náročnosti splnenia potrebných úloh v ňom [13].

Na toto bol pred začiatkom testovania brán zreteľ a patrične k tomu boli formulované otázky, prezentovanie úloh a sledovanie reakcií testujúcich užívateľov.

Kapitola 7

Záver

Cieľom tejto bakalárskej práce bol návrh, implementácia a testovanie informačného systému pre rezerváciu a výdaj prípravkov používaných na kalibráciu elektrónových mikroskopov. Výsledný systém je však možné využiť aj v iných odvetviach priemyslu, pre všeobecný výdaj a a sledovanie prípravkov, nástrojov prípadne aj menších zariadení.

Pre udržanie kvality a plynulosti výrobného procesu je dôležité aby mali zamestnanci k dispozícii spoľahlivé nástroje. Zavedením tohoto informačného systému do výroby sa systém tiež stáva nástrojom, ktorý zamestnanci používajú pri svojej práci a tým pádom automaticky vzniká aj vplyv na chod výroby a nepriamo aj na kvalitu výsledného produktu. To si je pred nasadením potreba plne uvedomiť a starostlivo zvážiť všetky pozitíva, ale aj možné negatíva.

7.1 Nasadenie

Implementovaný systém bol z časových dôvod nasadený iba čiastočne. Webová aplikácia, *DispatcherClient*, zatiaľ nebola plne nasadená pretože, je potrebné ju integrovať do ďalšej aplikácie používanej vrámci spoločnosti FEI Czech Republic s.r.o. Ostatné aplikácie sú plne funkčné a používané bez väčších problémov. Pripomienky a návrhy od zamestnancov sú priebežne vyhodnocované a pokiaľ nie sú protichodné a neodporujú požiadavkám, ktoré na projekt na začiatku zadalo vedenie spoločnosti, tak sú do jednotlivých aplikácií aj zapracované.

7.2 Ďalšie možnosti vývoja

Návrh tohoto systému od začiatku zohľadňoval prípadnú jednoduchú rozšíriteľnosť v budúcnosti. Priestor pre ďalšie možnosti vývoja projektu je hlavne v spresnení správy rezervácií na základe nazbieraných dát o časovom využití jednotlivých prípravkov. To znamená pristupovať k rezerváciám jednotlivo a nie na základe globálne nastavených pravidiel.

Užitočné rozšírenie by predstavovalo pridanie podpory ďalších jazykov, hlavne češtiny. Implementácia viacjazyčného rozhrania by nepredstavovala veľké problémy, nakoľko použité technológie to vždy aspoň čiastočne podporujú. Úplná podpora viacjazyčnosti by však vyžadovala zásadnejšie zmeny systému. Obmedzenie predstavuje hlavne databáza, ktorá pre poskytovanie takejto funkcionality nebola navrhnutá. Týka sa to to hlavne tabuľky `Tools`, kde predstavujú problém stĺpce `Description` a `Name`. Riešením by napríklad mohlo byť pridanie tabuľky `ToolsTanslations`, tá by obsahovala záznamy v každom podporovanom

jazyku a cudzím kľúčom by odkazovala tabuľku `Tools`, kde by boli iba jazykovo neutrálne stĺpce. Možných prístupov však existuje viac [12].

Čo sa týka voľby hardvéru, bolo by lepšie na začiatku projektu zvoliť relé s nastaviteľným hardvérovým časovačom a potom ho ovládať pomocou softvéru v rámci tohto intervalu. Zvýšilo by to bezpečnosť celého systému a značne uľahčilo implementáciu ovládania zámkov.

Ďalej je nevyhnutné vypracovať úplnú technickú dokumentáciu o celom systéme. Tak aby jeho modifikovanie, ale aj spravovanie bolo jednoduché a mohol túto činnosť v prípade potreby prebrať iný zamestnanec firmy. Tento dôležitý krok ešte nebol úplne dokončený.

7.3 Zhodnotenie

Systém bol navrhnutý pre maximálnu jednoduchosť, tak aby splnil svoj účel a vyžadoval minimálnu interakciu s užívateľom. Výhodou takéhoto prístupu je vysoká efektivita výdaja a nízka réžia s pohľadu užívateľov. Má to však aj jednu nevýhodu. Systém je silne závislý na správnosti vložených dát. Funkcie, ako automatické vracanie prípravkov, blokovanie prípravkov s neplatnými revíziami alebo upozorňovanie na blížiace sa revízie budú pri nesprávnych vstupných dátach poskytovať zavádzajúce údaje.

Celkovo je možné projekt hodnotiť ako úspešný. Systém oproti svojmu predchodcovi poskytuje potrebnú funkcionalitu a vyšší komfort pre zamestnancov.

Literatúra

[1] Adam, K.; Dorota, H.: *Automated Defect Prevention: Best Practices in Software Management*. Wiley-IEEE Computer Society Press, 2007, ISBN 0-470-04212-5, 70 s.

[2] Alena, B.; Jarmila, P.; Luboš, P.: *Základy softwarového inženýrství - materiály ke cvičení*. Praha: Oeconomica, prvé vydanie, 2007, ISBN 978-80-245-1270-9.

[3] Arlow, J.; Neustadt, I.: *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*. Addison-Wesley Professional, druhé vydanie, 2005, ISBN 978-0321321275.

[4] Davenport, T. H.: *Process Innovation: Reengineering Work Through Information Technology*. Harvard Business Review Press, 1992, ISBN 978-0875843667.

[5] Fowler, M.: *GUI Architectures*. [Online; cit. 25.04.2017].
URL https://martinfowler.com/eaaDev/uiArchs.html

[6] Fowler, M.: *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Professional, tretie vydanie, 2003, ISBN 0-321-19368-7.

[7] Ian, S.: *Software Engineering*. Pearson, desiate vydanie, 2015, ISBN 978-0133943030.

[8] Jiantao, P.: *Software Testing*. [Online; cit. 29.04.2017].
URL https://users.ece.cmu.edu/~koopman/des_s99/sw_testing/

[9] Jindřich, K.; Ludmila, K.: *Modelování dat v informačních systémech*. Ekopress, 2012, ISBN 978-80-86929-49-1.

[10] Jon, G.; Brad, W.; David, M.; aj.: *Professional ASP.NET MVC 5*. John Wiley & Sons, 2014, ISBN 978-1-118-79475-3.

[11] Rick, A.: *Bundling and Minification*. [Online; cit. 3.05.2017].
URL https://docs.microsoft.com/en-us/aspnet/mvc/overview/performance/bundling-and-minification

[12] Shantanu, K.: *How to Design a Localization-Ready System*. [Online; cit. 1.05.2017].
URL http://www.vertabelo.com/blog/technical-articles/data-modeling-for-multiple-languages-how-to-design-a-localization-ready-system

[13] Steve, K.: *Don't Make Me Think: A Common Sense Approach to Web Usability*. New Riders, druhé vydanie, 2005, ISBN 978-0321344755.

[14] Taylor, M.; Rick, A.: *Razor syntax*. [Online; cit. 23.04.2017].
URL https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor

[15] Šmíd Vladimír: *Pojem informačního systému*. [Online; cit. 28.04.2017]. URL http://www.fi.muni.cz/~smid/mis-infsys.htm

Prílohy

Príloha A

Zoznam knižníc tretích strán

- Bootstrap
  http://getbootstrap.com/
- Bootstrap 3 Datepicker
  https://eonasdan.github.io/bootstrap-datetimepicker/
- Chosen
  https://harvesthq.github.io/chosen/
- jQuery
  https://jquery.com/
- jqPlot
  http://www.jqplot.com/
- PagedList
  https://www.nuget.org/packages/PagedList
- toastr
  https://www.nuget.org/packages/toastr
- MVC Toastr Flash
  https://www.nuget.org/packages/RedWillow.MvcToastrFlash

Príloha B

Zoznam skratiek

A	Ampér
AC	Alternating Current
AJAX	Asynchronous JavaScript and XML
CSS	Cascading Style Sheets
DC	Direct Current
EAN	International Article Number, European Article Number
EF	Entity Framework
ERD	Entity-Relationship Diagram
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IS	Informačný Systém
JSON	JavaScript Object Notation
LINQ	Language Integrated Query
MVC	Model–view–controller
MVVM	Model–view–viewmodel
napr.	napríklad
ORM	Object-relational Mapping
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
t. j.	to jest
UI	User Interface
UML	Unified Modeling Language
URL	Uniform Resource Locator
USB	Universal Serial Bus
V	Volt
WPF	Windows Presentation Foundation
XML	Extensible Markup Language

Príloha C

Obsah priloženého CD

- súbor s popisom inštalácie systému
- zdrojové súbory implementovaného systému
- zdrojové súbory technickej správy
- ovládače pre USB relé kartu